Test Data Decompression for Multiple Scan Designs with Boundary Scan

Janusz Rajski, *Member, IEEE*, Jerzy Tyszer, *Senior Member, IEEE*, and Nadime Zacharia

Abstract—The paper presents an efficient scheme to compress and decompress in parallel deterministic test patterns for circuits with multiple scan chains. It employs a boundary-scan-based environment for high quality testing with flexible trade-offs between test data volume and test application time while achieving a complete fault coverage for any fault model for which test cubes are obtainable. It also reduces bandwidth requirements, as all test cube transfers involve compressed data. The test patterns are generated by the reseeding of a two-dimensional hardware structure which is comprised of a linear feedback shift register (LFSR), a network of exclusive-or (XOR) gates used to scramble the bits of test vectors, and extra feedbacks which allow including internal scan flip-flops into the decompressor structure to minimize the area overhead. The test data decompressor operates in two modes: pseudorandom and deterministic. In the first mode, the pseudorandom pattern generator (PRPG) is used purely as a generator of test vectors. In the latter case, variable-length seeds are serially scanned through the boundary-scan interface into the PRPG and parts of internal scan chains and, subsequently, a decompression is performed in parallel by means of the PRPG and selected scan flip-flops interconnected to form the decompression device. Extensive experiments with the largest ISCAS' 89 benchmarks show that the proposed technique greatly reduces the amount of test data in a cost effective manner.

Index Terms—Boundary scan, built-in self-test, design for testability, reseeding of LFSRs, multiple scan chains, scan-based designs, test data decompression.

1 Introduction

Unprecedented proliferation of very large scale integrated (VLSI) circuits is accompanied by quickly increasing cost of their testing, which, in many areas, becomes a major, or even predominant, component of the overall cost associated with manufacturing and shipping of integrated circuits. This is largely because many test system technologies continue to follow traditional methodologies in which test vectors are applied and test responses are analyzed by means of expensive external testing equipment. With rising off-chip frequencies and chip pad counts being highly unbalanced by increasing complexity of circuits, there is an indispensable need to use lower-priced test systems enhancing the traditional test methods in terms of reduced dependency on physical probes, at-speed test capabilities, increased test portability, and significantly reduced test costs.

In built-in self-test (BIST) approach, now a rapidly emerging means of testing, on-chip hardware both generates test patterns and evaluates output data [1], [2], [4]. One of the major advantages of BIST is its ability to operate at different levels of a circuit's architectural hierarchy. However, in order to invoke the BIST procedures and facilitate their correct execution at the board, module or system level, certain design rules must be applied. In 1990, a testing standard, known as the IEEE Standard 1149.1 or the IEEE Standard Test Access Port and Boundary-Scan Architecture, was adopted [14]. The basic architecture of boundary-scan is incorporated at the integrated circuit level and essentially consists of a protocol by which various test functions can be carried out.

Every chip designed according to the standard contains a boundary-scan Instruction Register and associated decode logic. It is used to set the mode of operation for selected data registers by means of boundary-scan instructions which place data registers between the Test Data In (TDI) and the Test Data Out (TDO) terminals. Two registers are always present: the Bypass Register and the Boundary Register. Several additional registers are allowed under the optional clause of the 1149.1, and they can be selected by sending the proper control sequences to an appropriate controller. The latter feature will be used in this paper to implement an efficient test data decompression scheme for multiple scan testable chips.

A test pattern generator which guarantees complete fault coverage while minimizing test application time, area overhead, and test data storage is essential for a successful BIST scheme. Many different generation schemes have been proposed to accomplish various trade-offs between these parameters. The solutions range from exhaustive [5], [19], [20] and pseudorandom [4] techniques, which do not use any storage but take a long application time and often do not detect some faults, to deterministic automatic test pattern generation (ATPG)-based techniques [3], [6], [7] that may require significant test data storage but achieve complete fault coverage in a relatively short time.

- *J. Rajski is with Mentor Graphics Corporation, 8005 SW Boeckman Road, Wilsonville, OR 97070. E-mail: rajski@wv.mentorg.com.*
- *J. Tyszer is with the Institute of Electronics and Telecommunications, Poznan University of Technology, ul. Piotrowo 3a, 60-965 Poznan, Poland. E-mail: tyszer@et.put.poznan.pl.*
- *N. Zacharia is with Matrox Graphics Inc., 1055 St. Regis Blvd., Dorval, Quebec H9P 2T4, Canada. E-mail: nadime.zacharia@matrox.com.*

Manuscript received 2 Sept. 1997.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 105571.

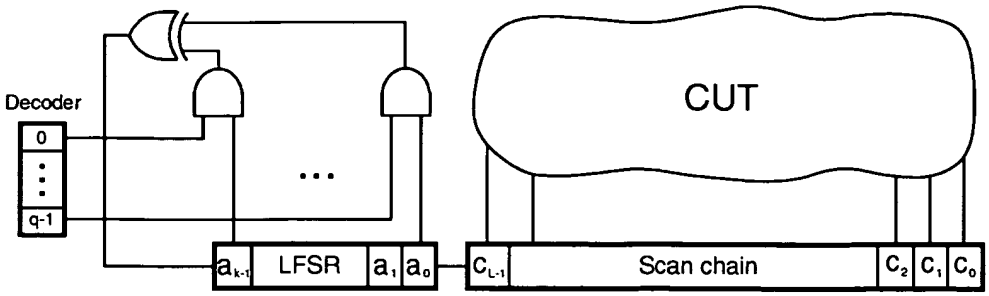

Fig. 1. Test model.

Mixed-mode test pattern generation is an attractive alternative to the above scenarios [8], [11], [13]. It uses pseudorandom patterns to cover easy-to-test faults and, subsequently, deterministic patterns to target the remaining hard-to-test faults. As opposed to other approaches, such as test point insertion [17], [18], mixed-mode techniques can reach complete fault coverage without imposing circuit modifications and causing performance degradation. Moreover, it is possible to obtain different trade-offs between test data storage and test application time by varying the relative number of deterministic and pseudorandom patterns. However, the overall efficiency of BIST scheme resting on mixed-mode generators strongly depends on methods employed to reduce the amount of test data.

There are two main approaches to reduce the quantity of test data:

1) reduction of the number of deterministic patterns by using test generation techniques that target as many single faults as possible with a single pattern, and
2) compression of deterministic test cubes by exploiting the fact that they frequently feature a large number of unspecified positions.

One of the methods to compress test cubes is based on the reseeding of linear feedback shift registers and was originally proposed in [13]. A comprehensive analysis of this scheme, as well as new reseeding scenarios based on multiple polynomial linear feedback shift registers (MP-LFSRs), were provided in [9], [21]. Using this method, a concatenated group of test cubes with a total of s specified bits is encoded with approximately s bits specifying a seed and a polynomial identifier. The content of the MP-LFSR is loaded for each test group and has to be preserved during the decompression of each test cube within the group. Accordingly, the implementation of the decompressor may involve adding extra flip-flops to avoid overwriting the content of the MP-LFSR during the decompression of a group of test patterns. In order to improve the encoding efficiency, an ATPG algorithm capable of generating test sets that minimize the storage requirements for the final encoded patterns has been recently proposed in [10].

This paper presents a new scheme to compress and decompress deterministic test vectors. The scheme is based on the concept of reseeding but uses variable-length seeds (Section 2). Consequently, the test patterns are generated by an LFSR loaded with seeds whose lengths may be smaller than the size of the LFSR. Allowing such "shorter" seeds yields high encoding efficiency even for test cubes with varying number of specified positions. Thus, concatenation of vectors is not required. The paper analyzes the efficiency of the scheme and presents a simple modular design of the decompression hardware in which the MP-LFSR shares some flip-flops with the scan to reduce the area overhead. This is in contrast to the former techniques, where the state of the decompressor could not be overwritten between applications of test cubes. Subsequently, in Section 3, an implementation of the proposed scheme in a multiple scan chain environment is shown. Finally, it is demonstrated in Section 5 how this two-dimensional decompressor structure can be tailored to boundary-scan architectures.

Numerous experiments with the largest ISCAS benchmarks are also reported to demonstrate that the scheme significantly reduces the amount of test data in a cost effective manner. Furthermore, dynamic compaction algorithms, which have been used by the ATPG process to support the scheme, are presented in Section 4. These algorithms minimize the number of generated test patterns by targeting many single faults with one vector. The paper concludes with Section 6.

2 Variable-Length Reseeding

2.1 Basic Concepts

Consider a k-bit MP-LFSR associated with 2^q primitive polynomials and assume that it feeds an L-bit scan chain, as shown in Fig. 1. For a given feedback polynomial, the bits of the output sequence can be expressed as linear combinations of the variables $a_0, a_1, a_2, \ldots, a_{k-1}$, where a_i is the ith bit of the initial seed vector.

A test vector $C = (c_0, c_1, c_2, \ldots, c_{L-1})$, $c_i \in \{0, 1, x\}$, with s specified bits is generated by the MP-LFSR if the system of linear equations $c_i = f_i(a_0, a_1, a_2, \ldots, a_{k-1})$ is consistent for at least one polynomial. Such a system yields an encoding in the form of a seed vector and a polynomial ID. In order to compress all the test vectors, some properties must be satisfied by the MP-LFSR. These properties were analyzed thoroughly in [9] and [11] in which different trade-offs between the length of the LFSR and the number of polynomials to implement are described. In particular, it is shown that if s is the maximum number of specified bits in a test vector, an s-bit MP-LFSR with 16 polynomials can practically compress all the test vectors. The seed calculation process is computationally simple and is equivalent to solving a system of s linear equations [9], [10], [13], [22]. This system has

a solution with probability greater than 0.999999 provided a decompression LFSR of length greater than $s + 20$ is used [9]. The system of equations can be solved very efficiently using Gauss-Jordan elimination, as fast row operations can be achieved with bit-wise operations.

In order to regenerate an L-bit test vector, the corresponding seed is loaded into the MP-LFSR and the polynomial ID is decoded to select the proper feedback polynomial. The MP-LFSR is then clocked for L cycles to produce an L-bit sequence consistent with the initial test vector.

EXAMPLE 1. Let the feedback polynomial be of the form $h(x) = x^3 + x^2 + 1$. If the LFSR is to generate a test cube $\{xx1x01x\}$, where x denotes a "don't care" condition, then a corresponding seed can be determined by solving the following system of equations:

$$\begin{aligned} a_0 &= x \\ a_1 &= x \\ a_2 &= 1 \\ a_0 + a_2 &= x \\ a_0 + a_1 + a_2 &= 0 \\ a_0 + a_1 &= 1 \\ a_1 + a_2 &= x. \end{aligned}$$

It can be easily verified that the resulting seed is $(a_2, a_1, a_0) = (1, 1, 0)$, which will subsequently produce a test pattern 0111010.

If the content of the MP-LFSR is initially reset, then the seed can be loaded into the MP-LFSR by shifting only the bits starting from the least significant 1. For instance, as the seed calculated in the above example is $(a_2, a_1, a_0) = (1, 1, 0)$, it will need only two clock cycles to be loaded into the MP-LFSR. In general, if a k-bit seed can be expressed as $(a_{k-1}, a_{k-2}, \ldots, a_{k-n}, 0, 0, \ldots, 0)$, $0 < n \leq k$, then it can be loaded into the MP-LFSR by first resetting the content of the MP-LFSR and, then, by shifting the n-bit vector $(a_{k-1}, a_{k-2}, \ldots, a_{k-n})$. Since the number of bits required to specify the initial value of the MP-LFSR may vary from vector to vector, it appears that the concept of variable-length seeds has a potential to offer better encoding efficiency than that of fixed-size seeds. Clearly, if there are different encodings that can represent a given test vector, the encoding that minimizes the length of the seed is always selected.

2.2 Encoding Efficiency

The encoding efficiency of the scheme depends on the format used to store the test data. Since the seeds have variable lengths, some additional information must be stored to specify the current length of the seed. In the proposed test data format (Fig. 2), it is assumed that the seeds are first sorted in increasing order of their length. The format then consists of three fields: the *size bit* field indicates when to increase the size allocated to store the seed. When its value is 1, the size of the seed field allocated for the next encoding is increased by Δ bits. The *polynomial* field contains the polynomial number associated with the seed. Since the number of polynomials associated with the MP-LFSR is 2^q, q bits are allocated to the field. Finally, the *seed* field contains the seed with some extra 0s appended. These extra 0s

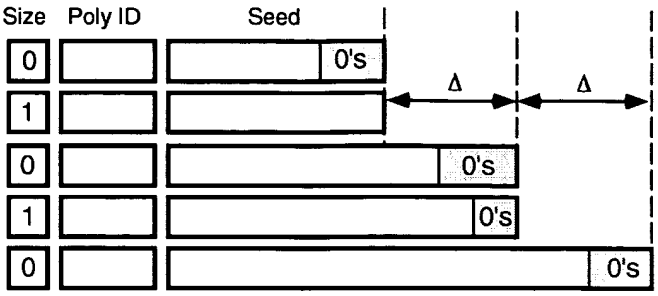

Fig. 2. Test data format.

are included so that the length of a seed field can be always expressed as $b + i\Delta$, $i = 0, 1, 2, \ldots$, where b is the size of the shortest seed. However, the value of the increment Δ should be chosen such that the number of extra zeros is kept at a minimum.

The encoding efficiency ξ, for a given vector, is defined as the number of specified bits it contains divided by the number of bits required to encode it, i.e.,

$$\xi = \frac{s}{1 + q + u + v}, \tag{1}$$

where s is the number of specified bits in the test vector, u is the expected length of the seed, and v is the average number of extra 0s appended per vector.

All parameters of (1) are known except u and v. The latter depends on Δ and the distribution of the number of specified bits in the test vectors. For uniform distributions, v can be shown to be very small and has little influence on ξ. On the other hand, the expected length of the seed u has a tremendous impact on ξ. The following discussion estimates u as a function of s and q. The obtained result will be then put into (1).

Let $P(n, s, q)$ be the probability that a test vector with s specified bits can be compressed with a seed of length n or shorter if there are 2^q polynomials associated with the MP-LFSR. Let us first focus on a single-polynomial case, i.e., assume that $q = 0$.

In order to derive $P(n, s, 0)$, let us consider the process of forming the equations to calculate the seed for a test vector with s specified bits. At every step, a new equation is formed and the probability that the system of equations remains consistent is calculated. The process starts with an empty set of equations and stops after s equations are formed. It can be described by a directed graph (Fig. 3) in which the vertices represent consistent systems of equations and the edges represent transitions that preserve the consistency of the system as a new equation is added. In the graph, vertex $X_{t,d}$ denotes a consistent system of t equations with rank d. Every edge is labeled with the probability of the corresponding transition. The transitions are functions of the rank d of the system.

As new equations are created, new vertices are added to the graph provided the system remains consistent. Note that two outgoing transitions that maintain the consistency of the system are always used for each node since a new equation can increase the rank d of the system or not. Let $\alpha(d)$ be the probability that the next equation does not

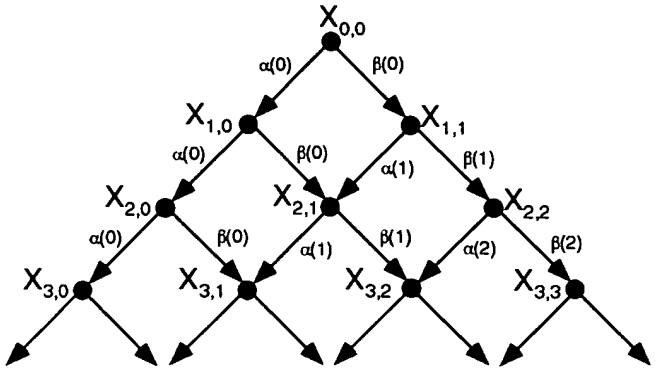

Fig. 3. Graph of equations.

increase the rank but preserves the consistency, and $\beta(d)$ be the probability that the next equation increases the rank d. For this case, the consistency is guaranteed.

To derive the transition probabilities $\alpha(d)$ and $\beta(d)$, let us consider the state $X_{t,d}$ and recall that it relates to a consistent system of t equations with rank d. The system of equations comprises t linear combinations of the initial seed vector variables. Since the rank is d, there are, in total, $2^d - 1$ different linear combinations that are dependent on those contained in the system represented by $X_{t,d}$. They can be obtained by systematic generation of all sums of already existing equations. These linear combinations all appear 2^{k-n} times in the m-sequence as $k - n$ positions are filled with constant 0s. If combinations comprising only these 0s are also taken into account, the total number of positions in the m-sequence that are linearly dependent on the t linear combinations contained in the system of equations is $(2^d - 1)2^{k-n} + 2^{k-n} - 1$. Since t of these positions are already included in the system of equations, the proportion of dependent positions in the m-sequence but not in the system of equations is therefore

$$\frac{2^{k-n}\left(2^d - 1\right) + 2^{k-n} - 1 - t}{2^k - 1} \approx 2^{d-n}, \tag{2}$$

for $2^k \gg 1$. It is assumed here that the positions are uniformly distributed in the m-sequence and that this proportion is valid even if the scan length is much smaller than the length of the m-sequence. It is also assumed that the length of the m-sequence is much larger than the number of specified bits in the test vector. The probability that the next equation formed increases the rank of the system is, therefore, $1 - 2^{d-n}$. For such a case, the consistency of the system is guaranteed. Therefore,

$$\beta(d) = 1 - 2^{d-n}. \tag{3}$$

If the next equation added does not increase the rank of the system, the consistency of the system is no longer guaranteed and depends on the constant part of the new equation. Since the specified bits of the test vector are assumed to be random, the probability that the system remains consistent is 1/2. This implies that

$$\alpha(d) = \frac{1}{2}2^{d-n} = 2^{d-n-1}. \tag{4}$$

It is important to note that since some positions in the m-sequence contain constant 0s or similar linear combinations, all states $X_{t,d}$ with $t \geq 0$, $d \geq 0$ have to be considered even if $t > 2^d - 1$. For instance, the state $X_{2,0}$ represents a consistent system of two equations having rank 0, i.e., the set of equations $\{0 = 0, 0 = 0\}$.

The probability that a seed of the form $(a_{k-1}, a_{k-2}, \ldots, a_{k-n}, 0, 0, \ldots, 0)$ exists is equal to the probability that the corresponding system of s equations is consistent. It is equal to the probability of reaching the vertices $X_{s,0}, X_{s,1}, X_{s,2}, \ldots, X_{s,s}$ as they represent all the possible consistent systems of s equations. For each path, the transition probabilities are multiplied together and the final result is the sum of these products. In particular, if $n = s$ and $n > 14$, calculations carried out in the graph yield $P(s, s, 0) \approx 0.61$.

The above discussion has been restricted to LFSRs with only one primitive polynomial. Let $P(n, s, q)$ be the probability that there exists at least one seed of length n or smaller such that an MP-LFSR associated with 2^q primitive polynomials can generate the corresponding test vector. In this case, calculating seeds for different primitive polynomials corresponds to statistically independent events and therefore $P(n, s, q)$ can be expressed as

$$P(n, s, q) = 1 - \left(1 - P(n, s, 0)\right)^{2^q}. \tag{5}$$

Let N be a discrete random variable that represents the actual length of the seed corresponding to a given test vector with s specified bits. The probability $P[N = n]$ can be expressed as

$$P[N = n] = P(n, s, q) - P(n - 1, s, q), \tag{6}$$

and the expected value of N can be calculated with

$$E(N) = u = \sum nP[N = n]. \tag{7}$$

Using the technique described earlier, numerous experiments were performed in order to determine $P[N = n]$. For the cases where n and s are larger than 14, it appears that $P[N = n]$ is virtually a function of $n - s$ and q. Fig. 4 shows $P[N = n]$ as a function of $n - s$ for $q = 0, 1, 2, 3$, and 4. Consequently, using data shown in Fig. 4, the expected value of N can be approximated by the following formula:

$$E(n) \approx s - q. \tag{8}$$

Finally, substituting $u = s - q$ and assuming that $v = 0$, the encoding efficiency ξ is given by

$$\xi = \frac{s}{1 + q + u + v} = \frac{s}{s + 1}. \tag{9}$$

Equation (9) demonstrates that, on the average, test vectors with s specified bits can be compressed with only $s + 1$ bits. For instance, test vectors with 50 specified bits can be compressed with 51 bits, yielding encoding efficiency better than 98 percent. Note that (8) indicates that, on the average, one polynomial would be sufficient to achieve this performance. However, a nonzero variance of N may occasionally lead to more than average usage of memory. In such cases, it is still better to employ more than one polynomial in order to guarantee the highest encoding efficiency possible.

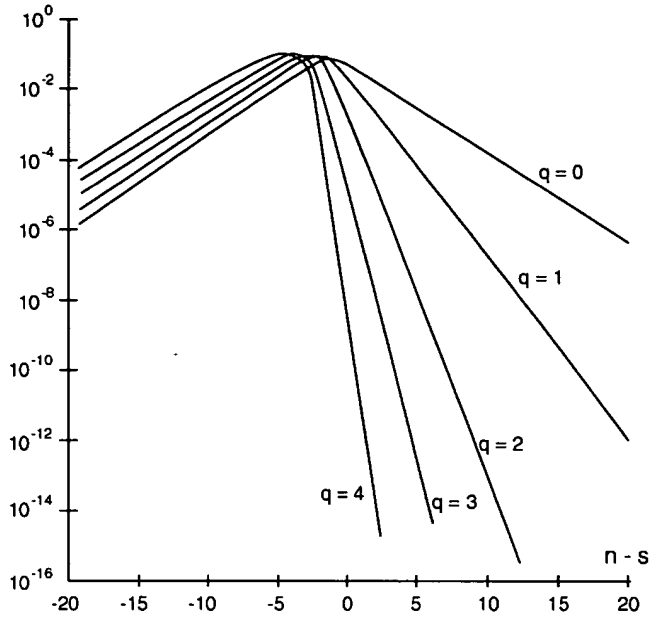

Fig. 4. $P[N = n]$ as a function of $(n - s)$.

2.3 Selection of Polynomials

The derivation of the encoding efficiency indicates that variable-length reseeding can be very efficient. However, it is important to recall that this derivation is valid when the whole m-sequence is considered. For a scan shorter than the m-sequence, the derivation is valid if it is assumed that the linear dependencies in the m-sequence are somewhat uniformly distributed. It has been shown [15] that the assumption may not be justified in general, especially if the scan length is only a small fraction of the length of the m-sequence. In such cases, the encoding efficiency may depend on the polynomials selected for reseeding.

Extensive Monte Carlo experiments have been performed to identify inherent properties of the polynomials that allow high encoding efficiency. All irreducible polynomials of degree 15 and 16 and a significant number of irreducible polynomials of degree 31 and 32 were analyzed. The primary objective of these experiments was to determine how the primitiveness of polynomials and the number of terms they feature can impact the encoding efficiency.

For every tested polynomial of degree k, 10,000 random test vectors of length $L = 10k$ with k specified bits were generated. From these vectors, the probability $P(k, k, 0)$ of finding a seed was measured and compared with the theoretical value of 0.61. Tables 1 and 2 present the average (avg) and worst (min) case values of P for each category of polynomials. The experimental results demonstrate that there is no significant difference between the average values of $P(k, k, 0)$ for primitive polynomials and nonprimitive polynomials. This implies that irreducible polynomials yield similar encoding efficiencies independently of their primitiveness. Moreover, the results show that the values of $P(k, k, 0)$ for primitive trinomials are considerably lower than the theoretical value of 0.61. This clearly indicates that trinomials may not guarantee high encoding efficiency. Finally, the minimum values of P point out that some polynomials with more than three terms do not permit high encoding efficiency either. The number of such polynomials, however, is rather small, as they do not have any effect on the average value of P. They are encountered only in Table 1, where the results for all irreducible polynomials of degree 15 and 16 are gathered.

In order to achieve encoding efficiency close to the theoretical value derived in Section 2.2, irreducible polynomials with five or more terms should be employed.

2.4 Hardware Implementation

The proposed decompression scheme can be implemented in hardware as a part of a built-in test strategy. The overall architecture consists of a test controller, a number of ICs with on-chip decompression hardware, and a memory unit. To apply one deterministic test vector, the test controller fetches the compressed test data from memory and transfers it through slow (serial) channel to the ICs. The on-chip test hardware decompresses the data it receives and applies the resultant test vector to the circuit under test. Recall that the polynomial IDs are appended to the seeds.

In order to minimize the area overhead, the design of the k-bit MP-LFSR used for reseeding is based on the LFSR structure used for pseudorandom test vector generation. As k depends on s_{max}, the maximum number of specified bits in a test vector, the MP-LFSR required for reseeding may be larger than the LFSR used for random test vector generation. In the following design, the extra flip-flops are shared with the flip-flops of the scan. The decompression hardware consists of the LFSR employed for random test vector generation with extra logic to share flip-flops with the scan and to handle seeds and polynomial IDs. Fig. 5 shows the design of the decompression hardware.

In order to share flip-flops with the scan, only one additional feedback is required. This feedback is taken from the scan in such a way that the total size of the MP-LFSR is greater than or equal to s_{max}, and is enabled by an AND-gate when the signal *Decompression* is high. To handle the seeds, a multiplexer placed at the input side of the MP-LFSR allows the seeds to be serially shifted. As only selected bits of the seeds are specified, a reset logic circuitry is included to clear the content of the whole LFSR before the seeds are loaded. For the polynomial IDs, a register stores the current polynomial ID, while some decoding logic selects the corresponding feedback taps.

The decompression hardware operates in one of three modes:

- Pseudorandom mode: The hardware generates pseudorandom test vectors (*Decompression* and *Load seed* signals are both 0),
- Shift mode: The seed and the polynomial ID are shifted into the decompression hardware (*Load seed* signal is 1); since the polynomial ID is appended to the seed, it is stored in the POLY ID register and the seed is transferred to the MP-LFSR,
- Decompression mode: The decompression hardware generates the vector associated with the seed and polynomial ID (*Decompression* signal is 1 whereas the *Load seed* is 0).

TABLE1
PROBABILITY OF ENCODING FOR ALL IRREDUCIBLE POLYNOMIALS OF DEGREE 15 AND 16

# Terms	Primitive			Nonprimitive		
	# Pols	avg	min	# Pols	avg	min
$P(15, 15, 0)$						
3	6	0.48	0.47	-	-	-
5	82	0.61	0.56	16	0.61	0.59
7	470	0.61	0.52	76	0.60	0.47
9	720	0.61	0.51	178	0.61	0.47
11	450	0.61	0.52	96	0.61	0.46
13	66	0.61	0.54	16	0.61	0.59
15	6	0.60	0.59	-	-	-
$P(16, 16, 0)$						
5	52	0.61	0.58	42	0.61	0.61
7	368	0.61	0.55	366	0.61	0.58
9	812	0.61	0.56	775	0.61	0.51
11	648	0.61	0.52	656	0.61	0.51
13	156	0.61	0.54	181	0.61	0.59
15	12	0.61	0.60	12	0.60	0.54

TABLE 2
PROBABILITY OF ENCODING FOR SOME IRREDUCIBLE POLYNOMIALS OF DEGREE 31 AND 32

# Terms	Primitive (31)			Primitive (32)			Nonprimitive (32)		
	# Pols	avg	min	# Pols	avg	min	# Pols	avg	min
5	218	0.61	0.52	22	0.61	0.60	16	0.60	0.59
7	232	0.61	0.60	28	0.61	0.60	24	0.61	0.60
9	285	0.61	0.60	36	0.61	0.60	37	0.61	0.60
11	237	0.61	0.60	27	0.61	0.60	29	0.61	0.60
13	269	0.61	0.60	32	0.61	0.60	31	0.61	0.60
15	256	0.61	0.60	45	0.61	0.60	29	0.61	0.60
17	254	0.61	0.59	35	0.61	0.60	32	0.61	0.60
19	257	0.61	0.60	30	0.61	0.60	33	0.61	0.60
21	252	0.61	0.60	33	0.61	0.60	31	0.61	0.60
23	270	0.61	0.60	39	0.61	0.60	35	0.61	0.60
25	245	0.61	0.60	34	0.61	0.60	35	0.61	0.60
27	238	0.61	0.60	26	0.61	0.60	40	0.61	0.60
29	196	0.61	0.50	35	0.61	0.60	31	0.61	0.60
31	8	0.59	0.58	14	0.61	0.60	18	0.61	0.50

The steps performed by the test controller to decompress and apply one test vector are as follows:

1) Reset the MP-LFSR including the flip-flops shared with the scan.
2) Switch to the shift mode.
3) Shift the seed with the polynomial ID being appended.
4) Switch to the decompression mode.
5) Decompress the test vector by applying L clock cycles to the MP-LFSR.
6) Switch to the functional mode.
7) Apply the test vector to the circuit under test.
8) Shift out the response to the test response analyzer.

The feedback polynomials used for reseeding are closely related to the polynomials used for pseudorandom vector generation. If $p(x)$ denotes one polynomial used for pseudorandom vector generation, then $h(x, i) = p(x)x^i + 1$ is the corresponding polynomial used for reseeding, where i is the number of flip-flops shared with the scan and the degree of $h(x, i)$ is equal to k. In order to further facilitate the design of the decompression hardware, it is of interest to identify such polynomials $p(x)$ that all their corresponding polynomials $h(x, i)$ would be good candidates for reseeding-based applications for $0 < k < 200$. As these polynomials could be used for all practical designs of the decompressor, the only parameter that would change for different circuits would be the position of the feedback tap taken from the scan chain.

Values of $P(k, k, 0)$ for each $h(x, i)$, where k is the degree of $h(x, i)$ and $h(x, i) = (x^{32} + x^{26} + x^{21} + x^{18} + x^7 + 1)x^i + 1$, for $0 < i < 169$, were measured by considering 10,000 vectors of size $L = 1{,}000$ with k specified bits. It appears that virtually every polynomial $h(x, i)$ yields values of $P(k, k, 0)$ very close to the theoretical value of 0.61, indicating good performance of all of these polynomials. Hence, they can be used for a wide range of designs by simply changing the position of the feedback in the scan. Similar experiments were repeated for 15 other polynomials with virtually the same results. Therefore, if the LFSR for pseudorandom generation is

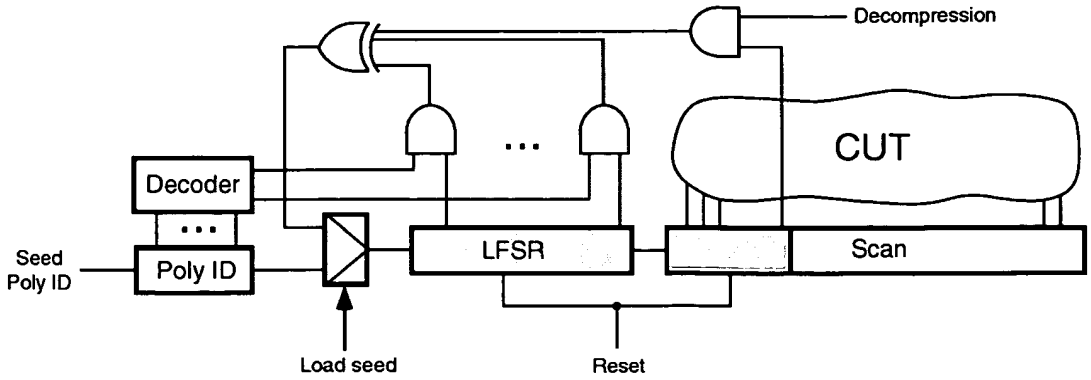

Fig. 5. Decompression hardware.

already implemented, the decompression hardware will require only a minimal area overhead consisting of q flip-flops to store the polynomial ID, some logic and a decoder to program the feedbacks of the MP-LFSR, one multiplexer to shift the seeds, and a reset circuitry for the flip-flops of the MP-LFSR.

2.5 Experiments

This section presents results of two experiments performed in order to measure the effectiveness of variable-length reseeding. The first experiment employed randomly generated test cubes, while the second experiment involved two industrial circuits.

In the first experiment, vectors of size $L = 1{,}000$ bits were generated with the numbers of specified bits being uniformly distributed between 20 and 200 with 10 vectors for each number of specified bits. The 1,810 generated vectors were then encoded with a 200-bit MP-LFSR with 16 polynomials (168 flip-flops were shared with the scan chain). The total number of specified bits in the test set was 199,100 bits while the number of bits required for the seeds was 191,271. Nine thousand fifty bits were used to provide information regarding polynomial IDs and size bits. Since the number of extra 0s was one, the total number of bits required to encode the test vectors was 200,322 bits. The resulting encoding efficiency (0.994) was then calculated as the ratio of 199,100 and 200,322.

The obtained result indicates that, on the average, a test vector with s specified bits can be compressed with only s bits. The results also point out that the number of extra 0s required to adjust the size of the seed field is minimal. This could be expected since the distribution of the number of specified bits in the vector was uniform.

In the second experiment, reseeding was applied to two industrial circuits CK1 and CK2 and the encoding efficiency for a subset of the test data (prior to merging) was measured. Table 3 describes the parameters of both circuits. Note that distributions of the numbers of specified bits in the test vectors are not uniform for these two cases.

The lower part of Table 3 presents the results when no extra 0s are assumed and Table 4 presents the result as a function of Δ when the extra 0s are taken into account. As can be seen, the optimal encoding efficiency is obtained when $\Delta = 1$ for CK1 and when $\Delta = 2$ for CK2. It is worth noting that the amount of extra 0s that was required is minimal.

For both circuits, the values of ξ are very close to 1.0, proving the high efficiency of the proposed scheme. Table 4 also illustrates the impact of Δ on the number of extra 0s required and on the encoding efficiency. As can be seen, a given value of Δ minimizes the number of extra 0s has a minimal impact on ξ. Furthermore, assuming that a 32-bit LFSR is already used for pseudorandom pattern generation, the scheme can be implemented with a negligible extra

TABLE 3
EXPERIMENTS WITH INDUSTRIAL CIRCUITS—NO EXTRA 0S

Parameters	CK1	CK2
Number of test patterns	544	1,424
Scan length	436	2,522
Distribution of number of specified bits	From 4 to 67	From 5 to 137
	Nonuniform	Nonuniform
Distribution of specified positions in test pattern	Nonuniform	Nonuniform
Total number of specified bits	11,259	55,607
Number of polynomials used	4	2
Degree of polynomials	68	162
Number of flip-flops shared with scan	36	130
Number of bits for seeds	9,931	54,245
Number of bits for polynomial ID and increase size	1,632	2,848
Total number of bits for seeds without extra 0s	11,563	57,093
Encoding efficiency	0.970	0.974

TABLE 4
ENCODING EFFICIENCY AS A FUNCTION OF Δ

CK1			CK2		
Δ	Extra 0s	ξ	Δ	Extra 0s	ξ
1	275	0.951	1	1,098	0.955
2	315	0.948	2	935	0.958
3	536	0.931	3	1,550	0.948
4	785	0.912	4	2,131	0.940

hardware. Indeed, for CK1, the scheme requires two flip-flops to store the polynomial ID, about 12 AND-gates and a 2-to-11 decoder to program the feedbacks of the LFSR, one multiplexer to shift the seed, and, finally, a reset circuitry. Similarly, for CK2, the scheme requires one flip-flop to store the polynomial ID, about 11 AND-gates and a 1-to-10 decoder to program the feedbacks of the LFSR, one multiplexer to shift the seed and the reset circuitry.

3 TWO-DIMENSIONAL HARDWARE DECOMPRESSOR

As most design for testability schemes employ multiple scan chains to guarantee acceptable test application time, two-dimensional test pattern generators have to be used to load the scan chains in parallel [4], [12]. Similarly, in BIST environment which rests on the reseeding technique, test cubes for hard-to-test faults can be compressed as variable-length seeds of a two-dimensional test pattern generator [23].

This section proposes a two-dimensional hardware decompressor for multiple-scan chain designs with a built-in self-test structure similar to that of the STUMPS architecture. The objective of the design is to allow the decompression of test cubes with a large number of specified bits, while minimizing the area overhead. Hence, the decompressor is implemented by using the flip-flops of the PRPG as well as the flip-flops from the scan chains. Fig. 6 shows the structure of the decompressor for a circuit having four scan chains. It is assumed that the PRPG is composed of a k-bit LFSR (type I) with an XOR network to scramble the bits at the input of the scan chains. The XOR network included in this example consists of one two-input XOR gate per scan chain but the first one.

The decompressor is implemented by adding extra feedbacks from the scan chains and some multiplexers. The feedbacks are used to combine the PRPG with scan flip-flops to implement the decompressor. These feedbacks are fed to additional XOR gates between the flip-flops of the PRPG and are controlled by means of AND gates.

As shown in Fig. 6, a feedback from a scan chain can be fed to more than one site in the PRPG. These sites are selected in such a way that the performance of the test data decompressor

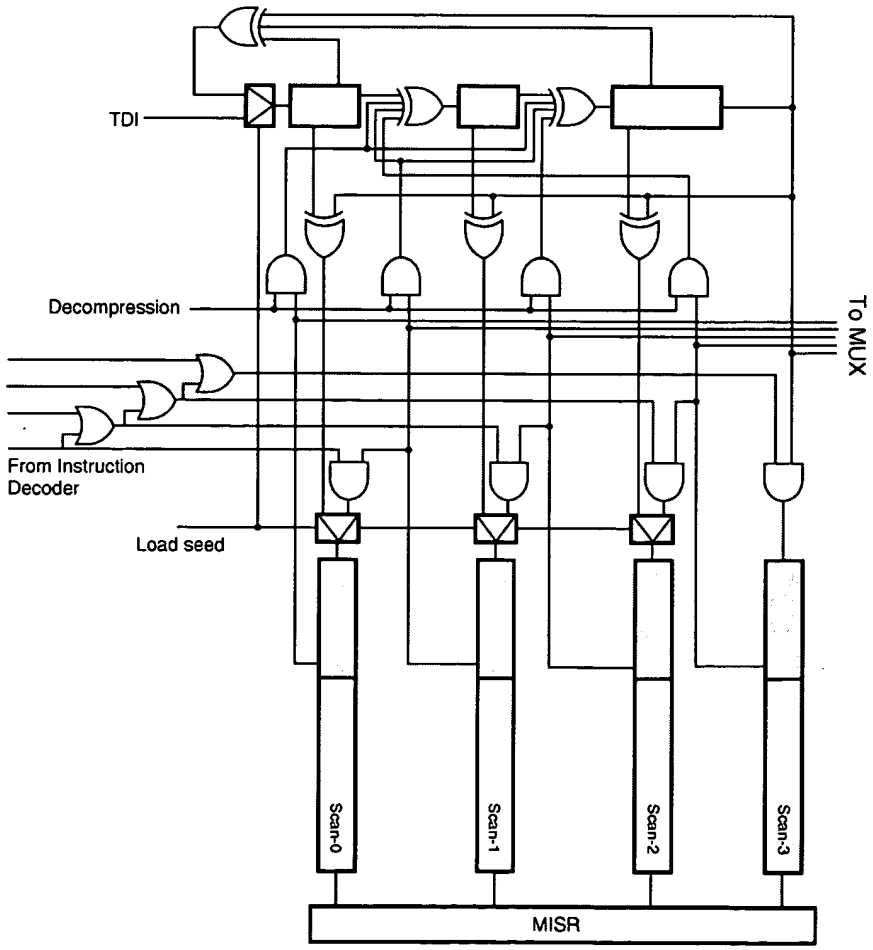

Fig. 6. Two-dimensional decompression hardware.

is optimized, i.e., the compression/decompression effectiveness is maximized. If the scan chains are numbered from 0 to $2^r - 1$, the feedback from scan number i is connected to positions $(i + 2^v) \bmod 2^r$, $v = 0, 1, 2, \ldots, r-1$, of the PRPG. The same feedback connection from a scan chain is provided to the input of the next scan chain (through a multiplexer). By adding these connections between the scan chains and placing a multiplexer in front of the PRPG, a serial path through the decompressor is created. It will be used to load the variable-length seeds.

The connections between successive scan chains are controlled using extra AND gates to facilitate filling of a part of the decompression structure with constant values. Therefore, when the bits of the seed are shifted in, the signals provided to the AND gates can force the value shifted into the scan chains to be 0 and, consequently some parts of the decompressor can be assigned to constant values of 0 and 1, depending on the presence of inverters between successive scan flip-flops. Thus, by using this approach, it is possible to load variable-length seeds without having to initialize the unused flip-flops prior to loading each seed.

As in the single scan chain case, the generator has two modes of operation: random and deterministic. In the random mode, the PRPG operates independently and generates pseudorandom patterns. In the deterministic mode, however, the extra feedbacks are enabled to implement the decompressor. By controlling the *Load seed* and *Decompression* signals, the variable-length seeds are loaded one by one and decompressed. The seeds are loaded by shifting the seed variables serially. Once the seed is loaded, decompression is performed in parallel through the XOR network.

Experiments were performed on the largest ISCAS '89 circuits with 1, 4, 8, 16, and 32 scan chains. The objective of these experiments was to measure the parameters of the scheme such as test application time, test data storage, area overhead, as well as to analyze the trade-offs between these parameters and the number of scan chains. The results are compared with those reported in [10] for circuits with a single scan chain.

For each circuit, 10K random patterns were generated by the PRPG working in the random mode and applied to the circuits in order to detect easy-to-test faults. After that, an automatic generator of deterministic test patterns with dynamic compaction (see Section 5) was used to produce test cubes required to achieve a complete stuck-at fault coverage. The resulting test cubes were then compressed into variable-length seeds of the decompressor of Fig. 6.

The decompressor was designed assuming a 32-bit PRPG with the feedback polynomial $h(x) = x^{32} + x^{29} + x^{11} + x^3 + 1$. The polynomial used to implement the PRPG should contain a sufficient number of feedback taps in order to reduce the probability of linear dependency, as was suggested in [15]. Consequently, the probability of not finding a seed for a given test cube will be minimized as well. The number of scan flip-flops used by the decompression structure was selected such that the total number of flip-flops in the decompressor is greater than $s_{max} + 20$, where s_{max} is again the maximum number of specified positions in any test cube. This insures that a seed can be obtained with probability greater than 0.999999 [9] if only one polynomial is employed.

TABLE 5
RESULTS FOR ISCAS '89 BENCHMARK CIRCUITS

Circuit	NS	LS	SFF	XOR	TD	CR
s9234	1	247	96	0	5,346	4.80
104	4	62	96	20	4,720	5.44
247	8	31	96	24	4,800	5.35
112	16	16	96	64	4,790	5.36
	32	8	96	64	4,968	5.17
s13207	1	700	192	0	5,877	20.96
176	4	175	192	12	5,784	21.30
700	8	88	192	32	5,908	20.85
183	16	44	192	48	5,840	21.09
	32	22	192	96	6,222	19.80
s15850	1	611	256	0	6,316	5.42
56	4	153	256	12	6,269	5.46
611	8	77	256	32	6,286	5.44
249	16	39	256	64	6,320	5.41
	32	20	256	128	8,612	3.97
s38417	1	1664	480	0	16,797	7.72
78	4	416	480	36	19,500	6.65
1664	8	208	480	32	16,858	7.69
472	16	104	480	64	16,844	7.70
	32	57	480	160	17,164	7.56
s38584	1	1464	224	0	3,996	19.05
52	4	366	224	32	3,901	19.51
1464	8	183	224	32	3,936	19.34
229	16	92	224	32	3,989	19.08
	32	46	224	64	4,037	18.86

The connection network between the scan chains and the PRPG was designed iteratively to reduce the number of XOR gates without deteriorating the compression effectiveness. The starting point was the configuration in which the feedback from scan number i is provided to positions $(i + 2^v) \bmod 2^r$, $v = 0, 1, 2, \ldots, r-1$, of the PRPG, where the scan chains are numbered from 0 to $2^r - 1$. At each iteration, the design was optimized by removing connections between the scan chains and the PRPG. The process was stopped when a simplification affected the compression effectiveness.

The first column of Table 5 shows the characteristics of the deterministic patterns that were generated. For each circuit, it lists the number of deterministic patterns required to achieve complete fault coverage, the size of each pattern in bits, and the maximum number of specified bits in a test cube, respectively. The remaining part of the table shows the results for each benchmark circuit after the seeds were obtained. The second column gives the number of scan chains inserted (NS). For each scan configuration, the table lists the length of the longest scan (LS), the number of scan flip-flops used to implement the decompressor (SFF), the number of extra two-input XOR gates required (XOR), the amount of test data that has to be stored after compression (TD), and the compression ratio (CR). The compression ratio was calculated by dividing the total amount of storage required to explicitly store the deterministic patterns, i.e., the product of first two quantities under the circuit name, by the amount of test data after compression (TD). Note

TABLE 6
COMPARISON BETWEEN THE PROPOSED SCHEME AND [10]
(AFTER 10K RANDOM PATTERNS)

Circuit	Scheme	# Pols	NC	Extra FFs	TD
s13207	VLR	1	176	32	5,877
	[10]	5	138 × 8	24	3,570
s15850	VLR	1	56	32	6,316
	[10]	5	134 × 8	46	6,528
s38417	VLR	1	78	32	16,797
	[10]	5	259 × 8	91	24,283

that the number of extra XOR gates does not include the XOR gates required to implement the PRPG.

Several conclusions can be drawn at this point. Clearly, no additional flip-flops are required to implement the decompression hardware, as it uses flip-flops from the PRPG and scan chains. Moreover, the number of scan flip-flops used (SFF) is substantial: more than 200 for large circuits. This number can also be interpreted as the number of extra flip-flops that would need to be inserted if the scan flip-flops could not be used. Thus, schemes that cannot use scan flip-flops, like those based on the concatenation of patterns, would require a large amount of extra hardware.

Based on Table 5, it can be seen that the amount of test data is small and comparable to the total number of specified bits in the deterministic test cubes, which yields high compression ratios. Also, there is a trade-off between the number of scan chains, the test application time, and the area overhead. Circuits with more scan chains require a shorter application time but need more XOR gates and multiplexers to implement the decompressor.

Table 6 compares the performance of the scheme proposed in this paper (referred to as VLR) with the one presented in [10] for some of the ISCAS circuits with a single scan chain. Recall that the scheme used in [10] is based on MP-LFSR reseeding and uses concatenation. Both schemes employ some particular kinds of dynamic compaction to generate test cubes targeting many single faults. For each circuit, the table reports the number of polynomials implemented by the decompressor (# Pols), the number of test cubes required to achieve complete stuck-at fault coverage (NC), the number of flip-flops needed to implement the test generator and the decompressor (Extra FFs), and the amount of test data after compression (TD).

As can be seen, the number of flip-flops inserted by VLR to implement the PRPG and decompressor is smaller for larger circuits. Also, VLR requires 15 percent less test data on the average. In [10], only the PRPG is used to implement the decompressor since concatenation precludes sharing scan flip-flops. When the circuit is small, e.g., s13207, a 32-bit PRPG is large enough for decompression. However, for larger circuits, i.e., s15850 or s38417, a larger PRPG has to be used which implies additional hardware. Clearly, the number of test patterns in [10] was increased to allow a smaller decompressor to be used.

4 GENERATION OF TEST CUBES

This section describes the test pattern generation scheme used to produce test cubes for hard-to-test faults. This strategy uses dynamic compaction to minimize the number of test cubes.

Prior to the generation of test cubes, a fault simulation is performed to identify the faults that fail to be detected by the pseudorandom patterns. The objective of the test cube generation is to detect the remaining faults with a minimum number of test patterns that result from the seeds calculated from the test cubes.

Test cubes are created by performing test generation on a selected fault. To adequately minimize the number of test patterns, it is necessary to perform test generation on additional faults searching for opportunities to extend the test cube to also detect those faults. Performing independent test generation on additional faults and then attempting to merge the patterns is not effective since it does not consider the "mergability" conditions when creating the patterns. A better way is to perform test generation on the remaining faults in the presence of the conditions set by the original test cube. Care must be taken to select faults that are most likely to be "mergable" or the test generation CPU time becomes unacceptable. The algorithm used to perform the generation of a test cube consisted of the following steps:

1) Select a target fault from the current fault list and create a test pattern that detects the fault. Extend the fault detectability of the pattern by attempting to sensitize all faults in the fanout-free network of gates in the fault detection path to any gate in the fault detection path. If a fault is sensitizable to any gate in the fault detection path, it will be highly likely to be detected. Reconvergent fault effects can sometimes result in the fault not being detected.
2) Add the stimulus of this pattern to the test cube and set the internal states of the circuit that result from the stimulus.
3) Randomly select a fault candidate for merging that lies outside the fanout free network of the target fault and any other fault that has been considered for the current test cube. Avoiding faults that share the same fanout free network is desirable since the detectability of many of these faults is already fixed. The fault candidate must also be consistent with the current state of the fault site and must have an unblocked path to an observe point considering the current internal states of the circuit.
4) Perform test generation for the fault candidate with the pattern extension described in Step 1. If the test generation is successful, add the stimulus of this pattern to the test cube and set the internal states of the circuit that result from the stimulus.

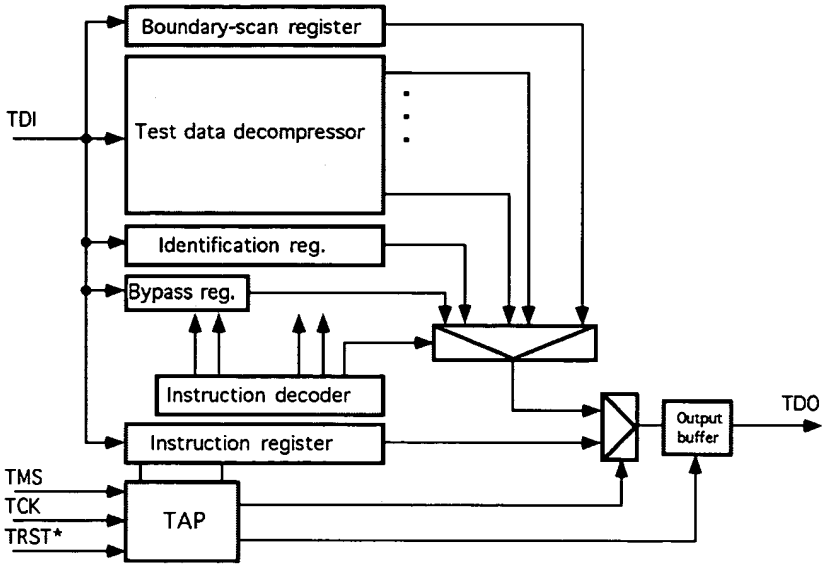

Fig. 7. Boundary-scan circuitry with the decompression module included as the user-defined test data registers.

5) Repeat Steps 3-4 until all possible fault candidates are used or until the number of unsuccessful attempts exceed a selected limit (default limit is 200).

A test cube only specifies the values on control points that were determined necessary to detect the target fault along with the additional faults. The final test cube is then used to calculate a seed that will result in a pattern that provides the values specified by the test cube. The complete test pattern that results from the seed is fault graded for the faults that remain in the fault list. Faults detected by the fault simulation are removed from the fault list and the test cube generation process is then iteratively repeated on the remaining faults until all faults are removed from the fault list.

5 Boundary Scan Environment

If integrated circuits are mounted in a board during testing, the reseeding-based test session can be still carried out in the boundary-scan environment, as shown in Fig. 7 and Fig. 8 [16]. The test data decompressor appears to the IEEE 1194.1 master as a collection serial scan registers, further referred to as decompression registers (Fig. 7). In fact, the decompression registers are constructed out of the same circuitry in such a way that each of them consists of the PRPG and (optionally) successive parts of the internal scan chains which can be selected by sending the proper sequence to the TAP controller (as defined by the standard, the TAP controller is a simple finite state machine driven by the Test Clock (TCK) and the Test Mode Select (TMS) pins). In addition to the standard IEEE 1194.1 instructions, several new instructions are added for accessing the decompressor. They include:

- SELPRPG: Select the PRPG only,
- SELSCAN0: Select the PRPG and first k_1 flip-flops from the first scan chain,
- SELSCAN1: Select the PRPG, first k_1 flip-flops from the first scan chain and k_2 flip-flops from the second scan chain,

and so forth. In general, the instruction SELSCANn allows for selecting the PRPG and flip-flops from $n + 1$ successive scan chains in such a way that the first k_i flip-flops of scan chain i are always taken, $i = 0, \ldots, n - 1$. Consequently, the decompression structure occurring on the die features several outgoing links (see Fig. 6) for serially shifting the seed bits out through the multiplexers and the TDO port.

In the testing mode, all TDIs and TDOs are daisy-chained from chip to chip, as illustrated in Fig. 8. The test session is organized as follows. First, a sequence of SELPRPG and/or SELSCAN instructions is shifted through the system in such a way that every chip receives its own content of the instruction register. These instructions place the appropriate parts of the decompressor structures between TDI and TDO. Fig. 8 shows a particular arrangement with the corresponding instructions used to configure this structure. For example, the fourth decompressor is using the PRPG only, while the last segment in the chain is employing all six scan chains to perform decompression of test data.

Second, the variable-length seeds are loaded. Each seed in the sequence has exactly the same length as the size of the corresponding decompression register. Signals from the instruction decoder are also used to properly initialize those parts of the scan chains which are not used in the current test phase. This is also done in order to avoid passing seed variables into those parts of the decompressor which must remain outside of the decompression registers. The corresponding circuitry, consisting of OR gates, is constructed in such a way that signal 1 applied to the input i automatically enables only scan chains with addresses lower than i.

Third, another new instruction, referred to as DECOMPRESS, is executed. While DECOMPRESS is in effect, the *Decompression* signal is active, thus allowing for decompression of loaded patterns and feeding the scan chains by means of the LFSR and the associated XOR network. This solution makes the following provision for the duration of the execution of the DECOMPRESS: An extra counter has

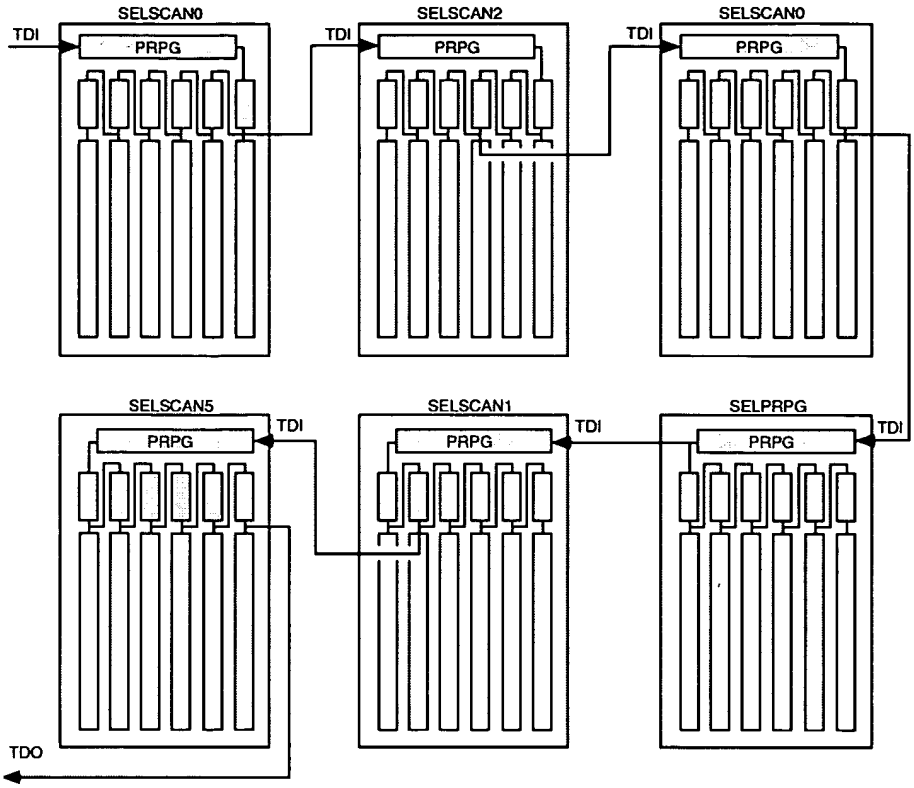

Fig. 8. Loading of seeds in the boundary-scan arrangement.

been added to every chip. When the DECOMPRESS is invoked, the counters are initialized to the size of the largest scan chain used on the corresponding chips. Subsequently, the counters count in descending order synchronously with the operation of feeding of the scan chains. After reaching state 0, the clock signal is disabled, a test pattern is applied to the circuit under test, and a test response is latched back into the scan chains. No further action is taken until all chips complete these operations.

Fourth, the next sequence of seeds can be loaded into the decompression structures, as described in Step 2, unless the entire system needs to be reconfigured, as indicated in Step 1, to allow loading of seeds of new sizes. Notice that during both phases, loading of seeds and, then, their decompression, the former test response is being gradually captured in the MISR, so, once the test is finished, the final signature can be shifted out for examination.

6 Conclusions

This paper proposes a new and very efficient scheme to decompress deterministic test vectors, to be used as part of a built-in test strategy using mixed-mode pattern generation. The scheme is based on the reseeding of an MP-LFSR and exploits variable-length seeds to encode the deterministic test vectors. The required hardware is very simple and introduces minimal area overhead, as it shares the LFSR used for pseudorandom test vector generation and some scan flip-flops. The scheme is compatible with scan designs and relies on mature automatic test pattern generation techniques to produce the test cubes. The scheme is also compatible with any fault models as long as the test cubes can be obtained.

The overall scheme, when used as a part of mixed-mode testing, offers many attractive trade-offs in terms of test data and test application time to achieve complete fault coverage. Moreover, it offers reduced bandwidth requirements as all the data transfers involve compressed data.

The hardware decompressor is well integrated with STUMPS designs. Furthermore, since the seeds are loaded serially, the approach is compatible with low-pin-count testers and boundary scan architecture. In the latter case, the entire decompressor structure is viewed as the user-defined test data registers accessible through the boundary-scan interface. The paper also introduces a dynamic compaction algorithm to support the schemes. The algorithm minimizes the number of test cubes by targeting multiple single faults with a single test cube.

The proposed schemes can be used in conjunction with test point insertion [18] and offer many attractive trade-offs between test data storage and test application time to achieve complete fault coverage. Whereas all the experiments were done with single stuck-at fault model, the scheme is compatible with any fault model as long as the deterministic test cubes can be generated.

Acknowledgments

The authors would like to thank Rami Mehio for providing very fast and reliable software to solve systems of linear equations, Robert C. Aitken for providing test cases, and J.A. Waicukauski for developing test cube generation procedures.

REFERENCES

[1] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "A Tutorial on Built-In Self-Test, Part 1: Principles," *IEEE Design and Test of Computers*, vol. 10, no. 1, pp. 73-82, 1993.

[2] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "A Tutorial on Built-In Self-Test, Part 2: Applications," *IEEE Design and Test of Computers*, vol. 10, no. 2, pp. 69-77, 1993.

[3] S. B. Akers and W. Jansz, "Test Set Embedding in Built-In Self-Test Environment," *Proc. Int'l Test Conf.*, pp. 257-263, 1989.

[4] P.H. Bardell, W.H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques*. John Wiley & Sons, 1987.

[5] Z. Barsilai, D. Coppersmith, and A.L. Rosenberg, "Exhaustive Generation of Bit Patterns with Applications to VLSI Self-Testing," *IEEE Trans. Computers*, vol. 32, no. 2, pp. 190-194, Feb. 1983.

[6] R. Dandapani, J.H. Patel, and J.A. Abraham, "Design of Test Pattern Generators for Built-In Test," *Proc. Int'l Test Conf.*, pp. 315-319, 1984.

[7] W. Daehn and J. Mucha, "Hardware Test Pattern Generators for Built-In Test," *Proc. Int'l Test Conf.*, pp. 110-113, 1981.

[8] C. Dufaza and G. Gambon, "LFSR-Based Deterministic and Pseudo-Random Test Patterns Generators Structures," *Proc. European Test Conf.*, pp. 27-34, 1991.

[9] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Trans. Computers*, vol. 44, no 2, pp. 223-233, Feb. 1995.

[10] S. Hellebrand, B. Reeb, S. Tarnick, and H.-J. Wunderlich "Pattern Generation for a Deterministic BIST Scheme," *Proc. Int'l Conf. Computer-Aided Design*, pp. 88-94, 1995.

[11] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *Proc. Int'l Test Conf.*, pp. 120-129, 1992.

[12] W.J. Hurd, "Efficient Generation of Statistically Good Pseudonoise by Linearly Interconnected Shift Registers," *IEEE Trans. Computers*, vol. 23, no. 2, pp. 146-152, Feb. 1974.

[13] B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs" *Proc. European Test Conf.*, pp. 237-242, 1991.

[14] C.M. Maunder and R.E. Tulloss, *The Test Access Port and Boundary-Scan Architecture*. IEEE CS Press, 1990.

[15] J. Rajski and J. Tyszer, "On Linear Dependencies in Subspaces of LFSR-Generated Sequences," *IEEE Trans. Computers*, vol. 45, no. 10, pp. 1,212-1,216, Oct. 1996.

[16] J. Rajski and J. Tyszer, "A Parallel Decompressor and Related Methods and Apparatuses," U.S. patent application, 1996.

[17] B.H. Seiss, P.M. Trousborst, and M.H. Schulz, "Test Point Insertion for Scan-Based BIST," *Proc. European Test Conf.*, pp. 253-262, 1991.

[18] N. Tamarapalli and J. Rajski, "Constructive Multi-Phase Test Point Insertion for Scan-Based BIST," *Proc. Int'l Test Conf.*, pp. 649-658, 1996.

[19] L.T. Wang and E.J. McCluskey, "Circuits for Pseudo-Exhaustive Test Pattern Generation," *Proc. Int'l Test Conf.*, pp. 25-37, 1986.

[20] H.-J. Wunderlich and S. Hellebrand, "The Pseudo-Exhaustive Test of Sequential Circuits," *IEEE Trans. Computer-Aided Design*, vol. 11, no. 1, pp. 26-33, 1992.

[21] S. Venkataraman, J. Rajski, S. Hellebrand, and S. Tarnick, "An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers," *Proc. Int'l Conf. Computer-Aided Design*, pp. 572-577, 1993.

[22] N. Zacharia, J. Rajski, and J. Tyszer, "Decompression of Test Data Using Variable-Length Seed LFSRs," *Proc. VLSI Test Symp.*, pp. 426-433, 1995.

[23] N. Zacharia, J. Rajski, J. Tyszer, and J.A. Waicukauski, "Two-Dimensional Test Data Decompressor for Multiple Scan Designs," *Proc. Int'l Test Conf.*, pp. 186-194, 1996.

Janusz Rajski received the MEng degree in electrical engineering from the Technical University of Gdansk, Poland, in 1973, and the PhD degree in electrical engineering from Poznan University of Technology, Poland, in 1982. From 1973 to 1984, he was a member of the faculty of Poznan University of Technology. In June 1984, he joined McGill University, Montreal, Canada, where he became an associate professor in 1989. In January 1995, he accepted the position of chief scientist at Mentor Graphics Corporation, Wilsonville, Oregon. His main research interests include design automation and testing of VLSI systems, design for testability (DFT), built-in self-test (BIST), and logic synthesis. He has published more than 80 research papers in these areas. He is co-author of *Arithmetic Built-In Self-Test for Embedded Systems* (Prentice Hall, 1997). He was co-recipient of the 1993 Best Paper Award for the paper on synthesis of testable circuits published in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. He has done contract work and has been a consultant to a number of companies in the area of testing.

Dr. Rajski was guest co-editor of the June 1990 and January 1992 special issues of *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* devoted to the 1987 and 1989 International Test Conferences, respectively. He is a member of the editorial board of the *Journal of Electronic Testing (JETTA)*, and an associate editor for *IEEE Design and Test* magazine, *IEEE Transactions on Computers*, and *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. He has served on technical program committees of various conferences and coordinated the topic of automatic test pattern generation and delay testing for International Test Conference. He is a co-founder and the general chair of the International Test Synthesis Workshop. He is a member of the IEEE.

Jerzy Tyszer received the MEng (Hons.) degree in electrical engineering from Poznan University of Technology, Poland, in 1981, the PhD degree in electrical engineering from the same university in 1987, and the Dr. Habilis degree in telecommunications from the Technical University of Gdansk, Poland, in 1994. From 1982 to 1990, he was a member of the faculty of Poznan University of Technology, Poland. In January 1990, he joined McGill University, Montreal, Canada, where was a research associate and adjunct professor. In 1996, he assumed the position of professor at the Institute of Electronics and Telecommunications of Poznan University of Technology, Poznan, Poland, where he carries on his work in the area of testing and synthesis of self-testable circuits. His main research interests include design automation and testing of VLSI systems, design for testability (DFT), built-in self-test (BIST), digital switching, and computer simulation of discrete event systems. He has published five books and more than 50 research papers in the above areas and is co-inventor of eight patents. He is coauthor of *Arithmetic Built-In Self-Test for Embedded Systems* (Prentice Hall, 1997). He has done contract work and has been a consultant in the area of testing to a number of companies and has served on technical program committees of various conferences. He is a senior member of the IEEE.

Nadime Zacharia received the BEng and the MEng degrees in electrical engineering in 1995 and 1998, respectively, from McGill University, Montreal, Canada. He is a member of the validation and test team at Matrox Graphics Inc. His current responsibilities include providing support in design for testability, test pattern generation, and fault simulation to various design groups within Matrox.